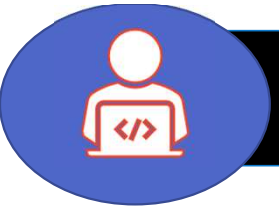
🔥 OOPS 🔥
Crash Course

Object Oriented Programming

తెలుగు లో JavaScript

Venkatesh Mogili
#WebGuru

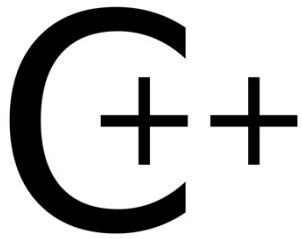


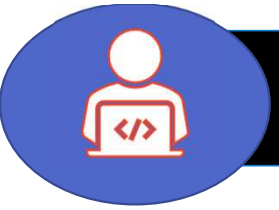
Contents

- OOPS Introduction
- Objects
- Classes
- Encapsulation
- Abstraction
- Inheritance
- Polymorphism

OOPS Introduction

- OOPS (Object Oriented Programming System or Paradigm)
- It aims to implement real-world scenarios like inheritance, hiding, polymorphism etc., in programming.
- **Smalltalk** is considered as the first truly object-oriented programming language.
- The popular object-oriented languages are Java, C#, PHP, Python, C++, etc.



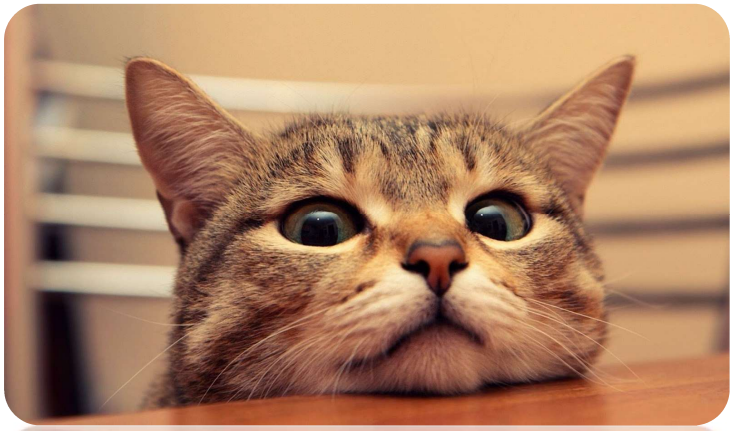


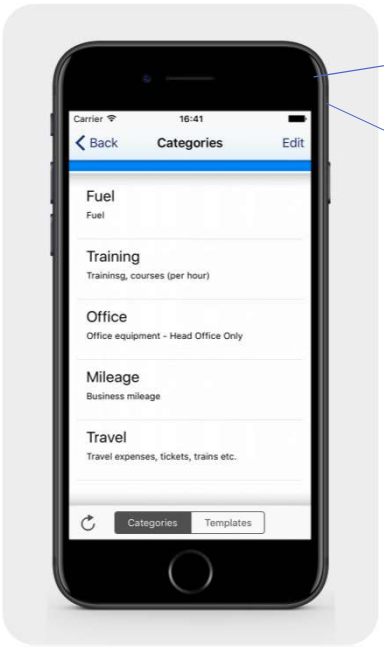
Contents

- ✓ OOPS Introduction
- ➢ **Objects**
- ➢ Classes
- ➢ Encapsulation
- ➢ Abstraction
- ➢ Inheritance
- ➢ Polymorphism

Objects

- Objects means simply real world entities such as mobile, laptop, bike, car, cat etc.,
- **Objects** have attributes/properties and actions/behaviors.

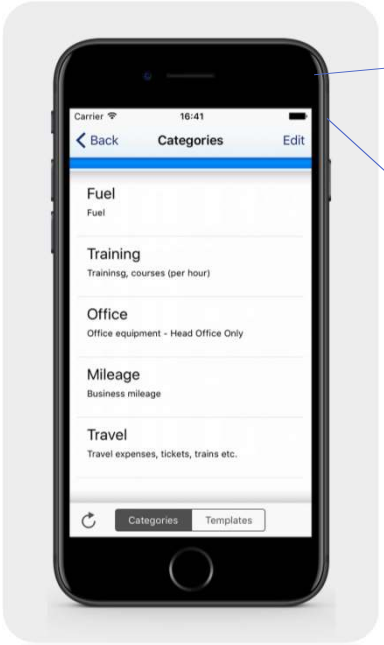




Attributes:
1.Black color
2.Apple Mobile

Actions:
1.Calling
2.Playing games
3.Taking pictures

Objects Example



Properties:
1.Black color
2.Apple Mobile

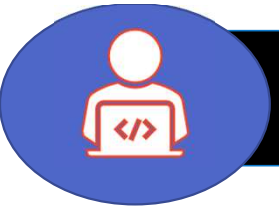
```
let mobile = {color:"black",name:"apple"};
```

Actions:
1.Calling()
2.Playing games()
3.Taking pictures()

```
function calling(){
  console.log('Call to Chinni');
}
```

```
function playPUBG(){
  console.log('Intro music🎵 -> Shoot enemy🔥 -> Close and Sleep😴');
}
```

```
function takePicture(){
  console.log('Smile Please 😃 -> Nice, Keep smiling like this');
}
```



Contents

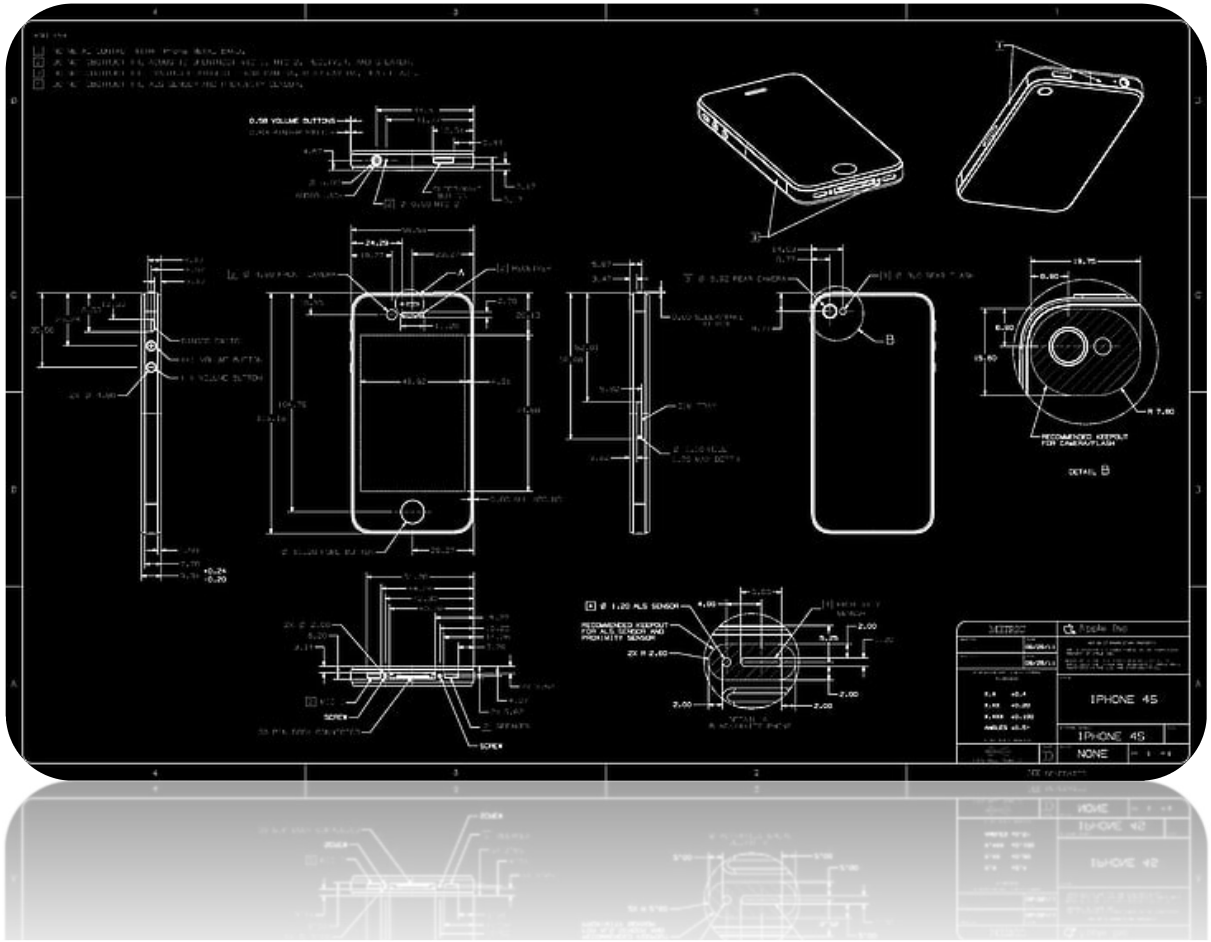
- ✓ OOPS Introduction
- ✓ Objects
- ➢ **Classes**
- ➢ Encapsulation
- ➢ Abstraction
- ➢ Inheritance
- ➢ Polymorphism

Class

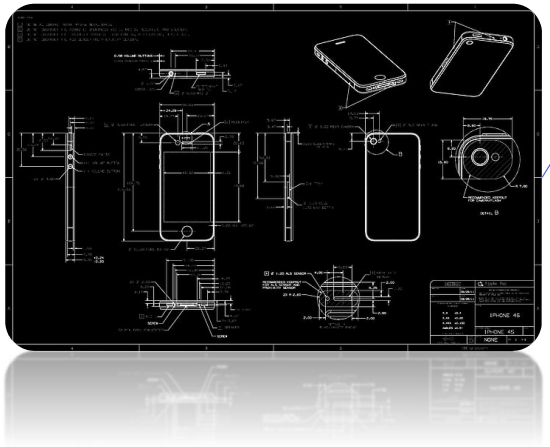
- Class is a blueprint for creating objects.
- We can create any number of objects with the same properties and actions by using Classes.

Example: Once Car class is created, then we can create any number of Cars with any name like Tesla, Audi, BMW with different features.

Class Example

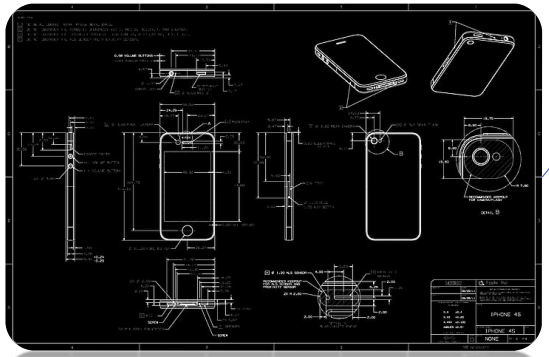


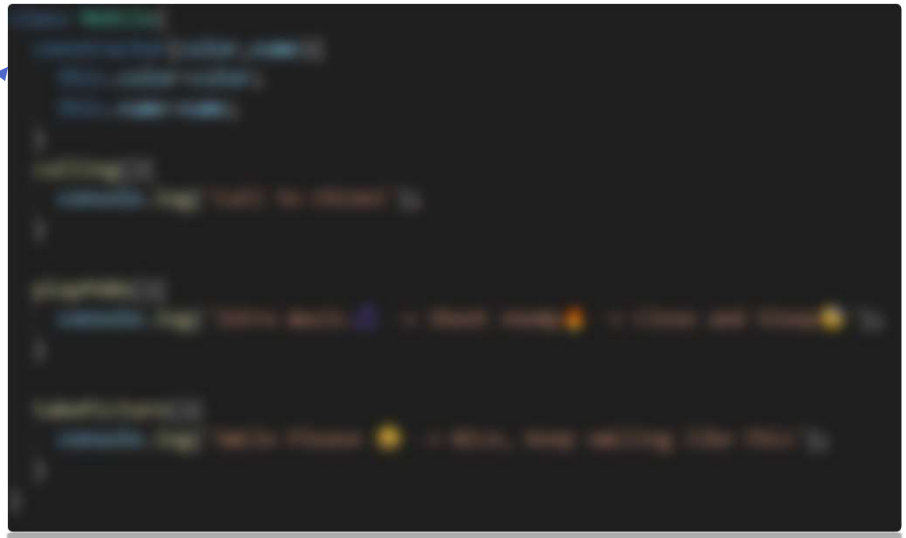
Class Example Scenario

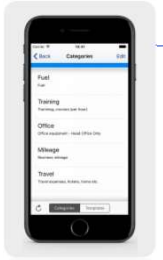


```
class Mobile{
  constructor(color,name){
    this.color=color;
    this.name=name;
  }
  calling(){
    console.log('Call to Chinni');
  }

  playPUBG(){
    console.log('Intro music🎵 -> Shoot enemy🔥 -> Close and Sleep😴');
  }

  takePicture(){
    console.log('Smile Please 😃 -> Nice, Keep smiling like this');
  }
}
```

Class and Object Creation

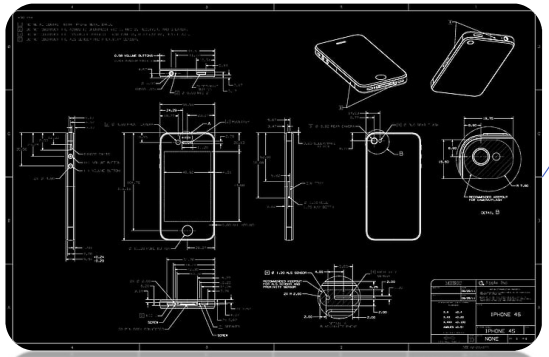


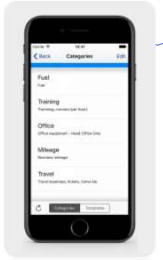




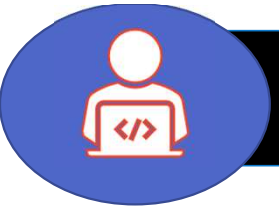
```
let apple = new Mobile('black','apple');
let redmi = new Mobile('silver','redmi');

console.log('Apple Mobile 📱 is ready 😃: ',apple);
console.log('Redmi Mobile 📱 is ready 😃: ',redmi);
```

Getting/Setting Attributes





```
// Properties/Attributes
console.log('What is the color of apple mobile?', apple.color);

// Actions/Functions
apple.calling(); //☎️
redmi.playPUBG(); // 🎮
apple.takePicture(); // 📷
```



Contents

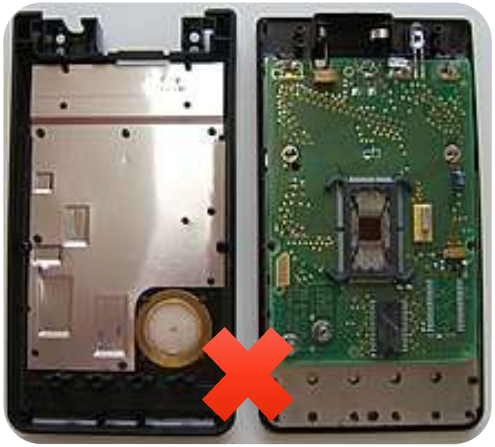
- ✓ OOPS Introduction
- ✓ Objects
- ✓ Classes
- ➢ **Encapsulation**
- ➢ Abstraction
- ➢ Inheritance
- ➢ Polymorphism

Encapsulation/Data Hiding

- **Encapsulation** is a functionality to hide the data in a single unit along with a functionality to protect information from outside.
- Data of an object should not be directly exposed.

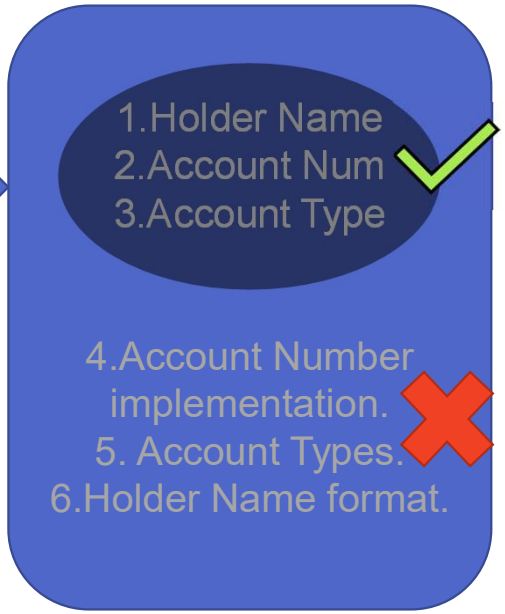
- Use var keyword to make data members private.
- Use setter methods to set the data and getter methods to get that data.

Example: The best **example** of **encapsulation** could be a calculator. We know that we can press 2+2 then = and see the result on display. We don't care about the internal parts like chip, electrical things how they are implemented etc.,



Encapsulation Examples

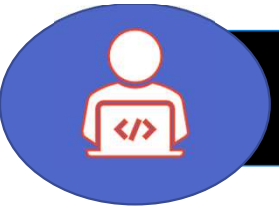
- More examples, more clarity 😃 Like, Share and Subscribe for supporting our channel and to help your friends 😃



Encapsulation Example Scenario

```
Name:  Venkatesh
Account No:  7031701841
Account Type:  Savings
```

```
Name:  Chinni
Account No:  2207524541
Account Type:  Current
```



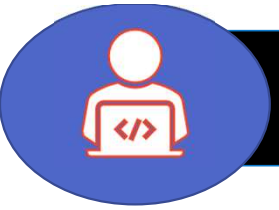
Interview Questions

- What exactly encapsulation means?
- **Encapsulation** is a method to hide the data in a single entity or unit along with a method to protect information from outside.
- Real life examples for encapsulation?

1. Capsule (which contains several medicines but bundles as a single unit)
2. Calculator (which contains several electrical devices but provides the simple interface to operate it)

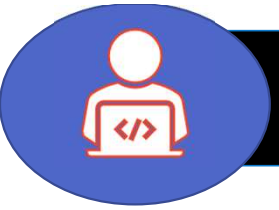
- Will it just hides the variables or methods as well?

✓ Yes, It should hide all the variables and it can hide any methods as well.



Contents

- ✓ OOPS Introduction
- ✓ Objects
- ✓ Classes
- ✓ Encapsulation
- ➢ **Abstraction**
- ➢ Inheritance
- ➢ Polymorphism

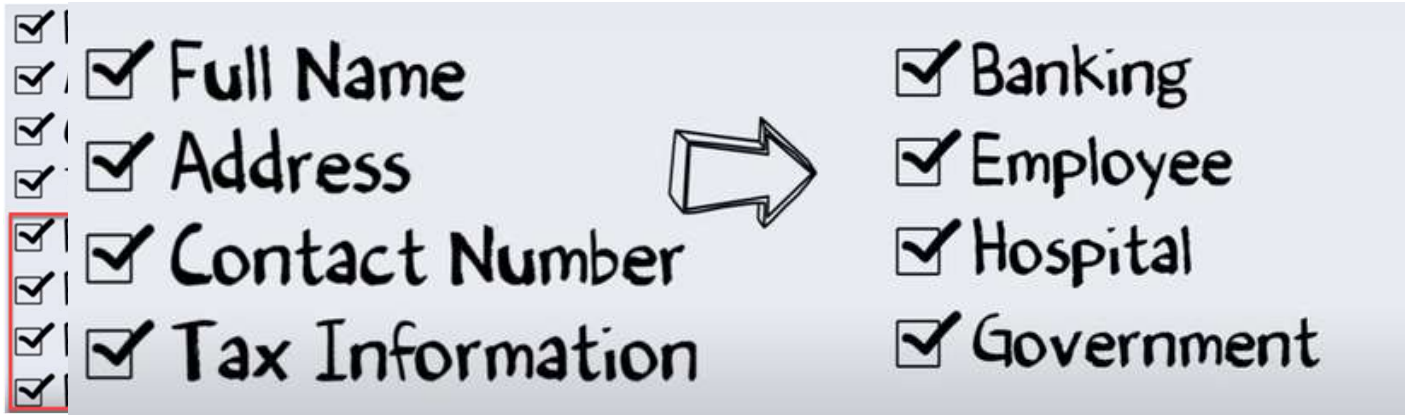


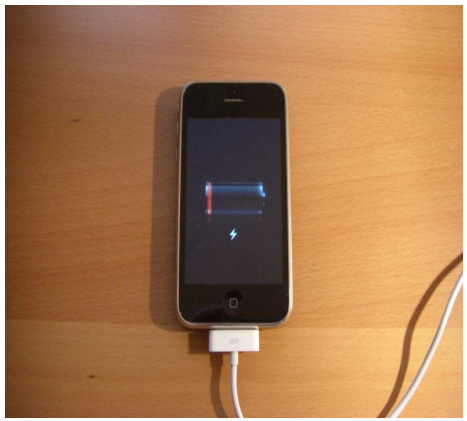
Abstraction

- **Abstraction** is the method that "shows" only essential attributes and "hides" unnecessary information. The main purpose of **abstraction** is hiding the unnecessary details from the users.

- Abstraction is selecting data from a larger pool to show only relevant details of the object to the user.

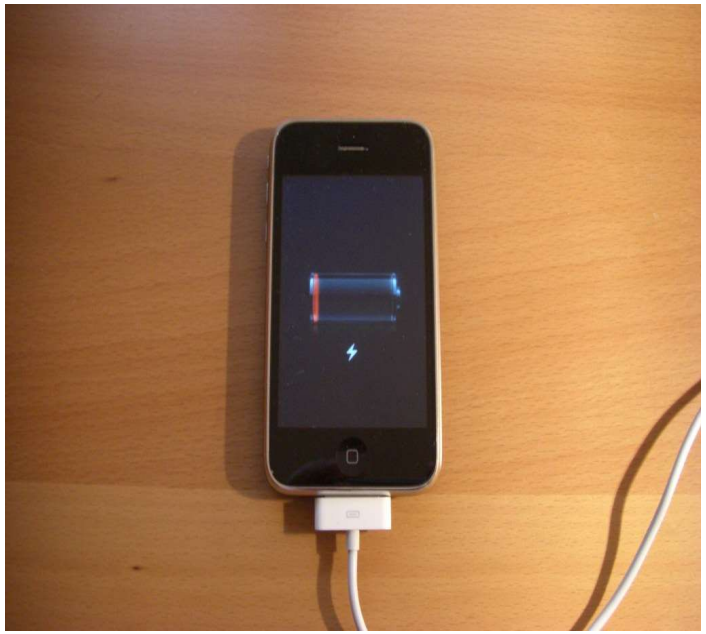
- It helps in reducing **programming** complexity and efforts.

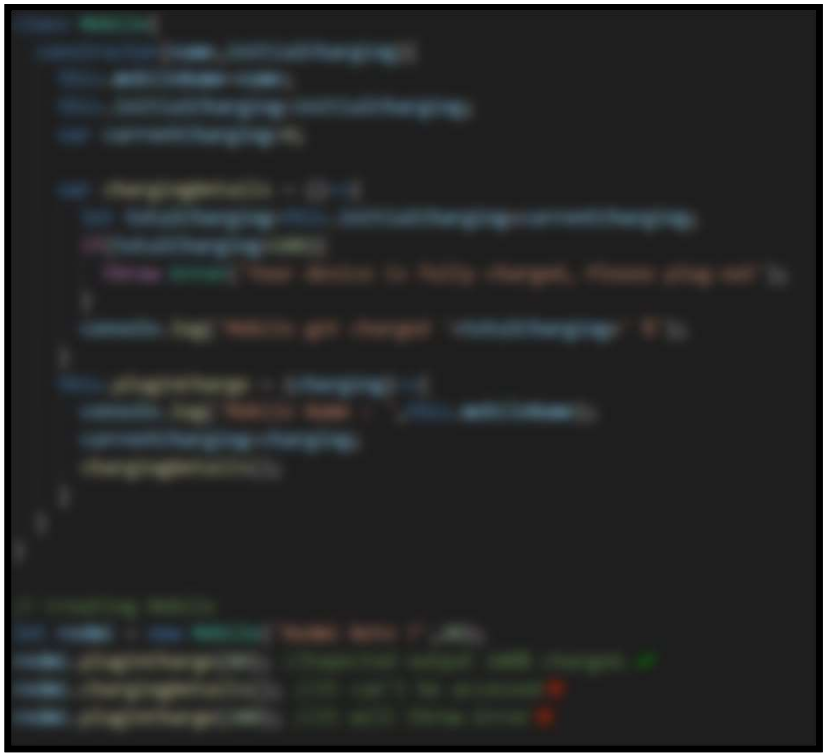




Example taken from Guru99.com

Abstraction Example Scenario





1

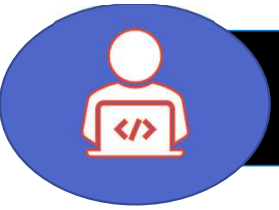
```
Mobile Name :  Redmi Note 7
Mobile got charged 100 %
```

2

```
redmi.chargingDetails is not a function
```

3

```
Mobile Name :  Redmi Note 7
▶ Uncaught Error: Your device is fully charged, Please plug-out
    at chargingDetails (oops.js:9)
    at Mobile.plugInCharge (oops.js:16)
    at oops.js:24
```



Interview Questions

- What exactly Abstraction means?
- **Abstraction** is the method of hiding the unwanted information.
- What is the difference between Encapsulation and Abstraction?
- **Abstraction** is the method of hiding the unwanted information.
  Whereas **encapsulation** is a method to hide the data in a single entity or unit along with a method to protect information from outside.

Interview Questions

- Question from Stack Overflow.

Difference between Encapsulation and Abstraction

Asked 7 years, 11 months ago Active 23 days ago Viewed 142k times

90

48

I had an interview today. I had a question from **OOP**, about the difference between **Encapsulation** & **Abstraction**?

I replied her to my knowledge that **Encapsulation** is basically to bind data members & member functions into a single unit called **Class**. Whereas **Abstraction** is basically to hide complexity of implementation & provide ease of access to the users. I thought she would be fine with my answer. But she queried, if the purpose of both are to hide information then what is the actual difference between these two? I could not give any answer to her.

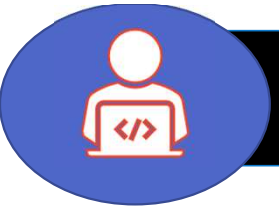
Before asking this question, I read other threads on StackOverFlow about the difference between these two **OOP**s concepts. But I am not finding my self in a position to convince the interviewer.

Can anyone please justify it with a simplest example?

Encapsulation

- Simplest example is a Phone with **Non-Removable battery** and with **Removable battery.**

Abstraction



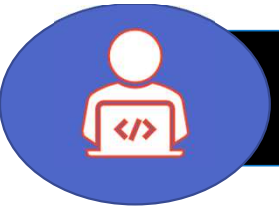
Interview Questions

- In Abstraction, We can provide the properties which can be accessed outside the class without setters and getter methods.
- But in encapsulation nothing can be accessed outside the class without setters and getter methods.

- Abstraction = Encapsulation + Non-Encapsulation 😃
- **Encapsulation = Encapsulation.**

- Both can hide the implementation details by just providing the getter and setter methods.

- As of now there is no clyster clear difference between both of them. We may get in future if there is more discussion happens in internet.



Contents

- ✓ OOPS Introduction
- ✓ Objects
- ✓ Classes
- ✓ Encapsulation
- ✓ Abstraction
- ➢ **Inheritance**
- ➢ Polymorphism

Inheritance

- Inheritance is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

Developer
Name
Designation

Learn()
Eat()
Sleep()
Coding()

Human
Name
Designation

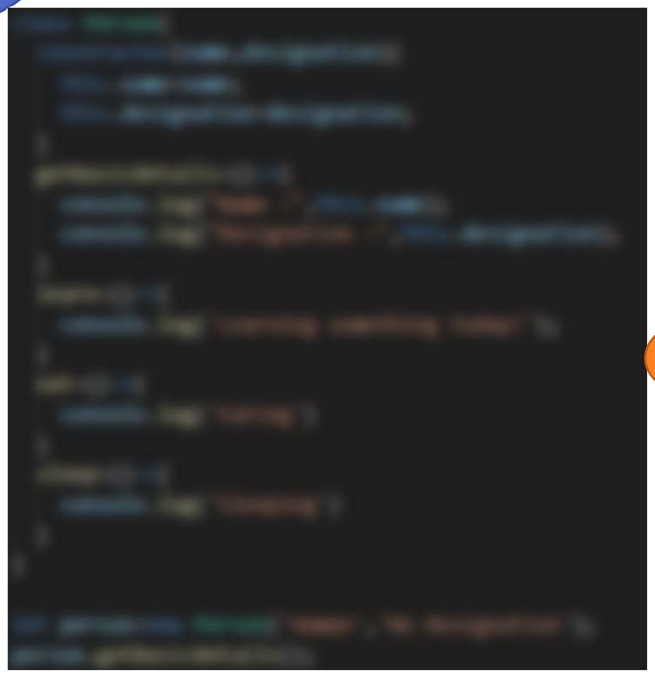
Learn()
Eat()
Sleep()

Dancer
Name
Designation

Learn()
Eat()
Sleep()
Dancing()

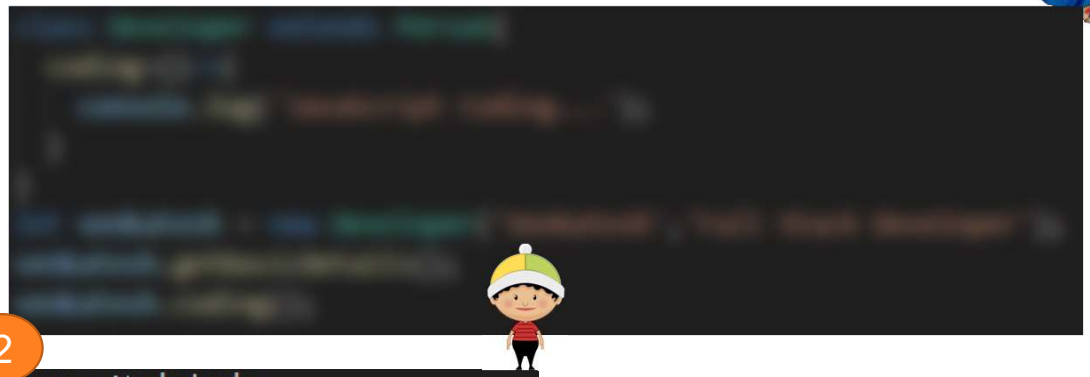
Example taken from Guru99.com

Inheritance Example Scenario



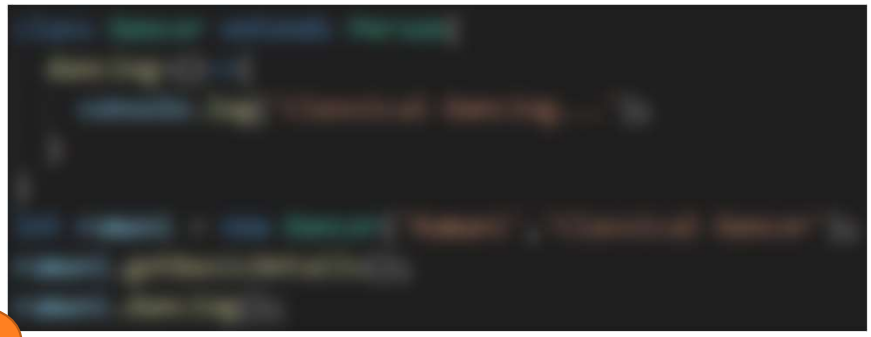
1

```
Name : Human
Designation : No Designation
```



2

```
Name : Venkatesh
Designation : Full Stack Developer
JavaScript Coding...
```



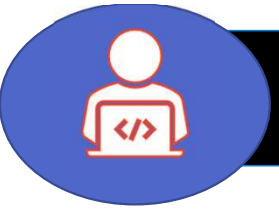
3

```
Name : Ramani
Designation : Classical Dancer
Classical Dancing...
```

Interview Questions

- Can you give an example for inheritance in React?
- Any components which extends React.Component class to create class components is one of the great example for inheritance.

- Can we extend more than once class?
- No, in **JavaScript**, a **class** cannot **extend** from **multiple classes**, which is also known as "**multiple** inheritance".

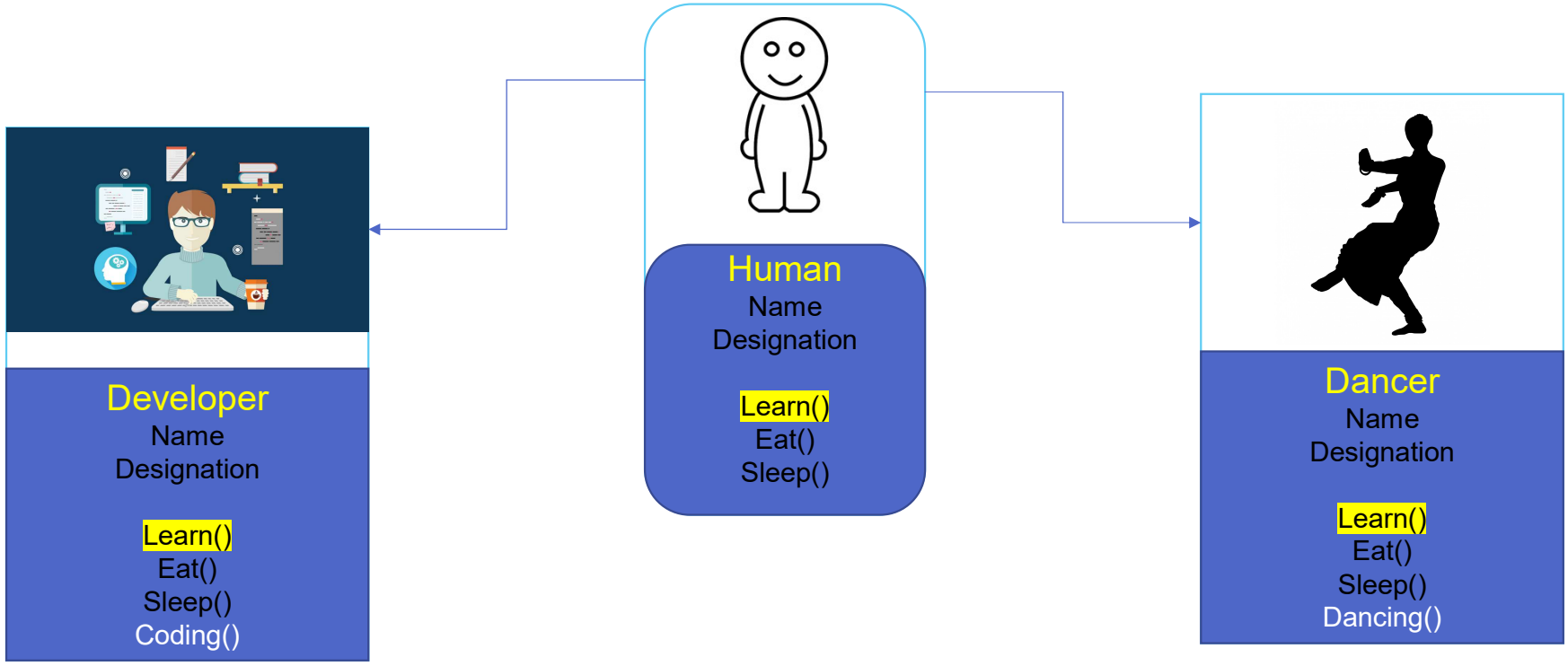


Contents

- ✓ OOPS Introduction
- ✓ Objects
- ✓ Classes
- ✓ Encapsulation
- ✓ Abstraction
- ✓ Inheritance
- ➢ **Polymorphism**

Polymorphism

- **Polymorphism** means having many forms.
- Example: same person can learn different things for different roles.



Example taken from Guru99.com

Polymorphism Example Scenario

```
[illegible]
```

```
[illegible]
```

2

```
Name : Venkatesh
Designation : Full Stack Developer
JavaScript Coding...
Learning OOPS Concepts in JavaScript!
```

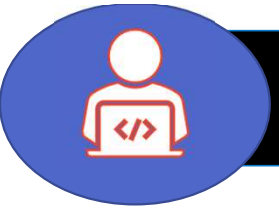
1

```
Name : Human
Designation : No Designation
```

```
[illegible]
```

3

```
Name : Ramani
Designation : Classical Dancer
Classical Dancing...
Learning Bharatha Natyam!
```

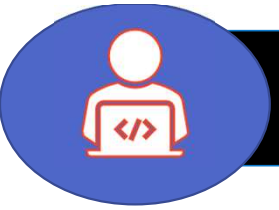


Interview Questions

- Can you give an example for polymorphism in React?
- Any react component should have render() method, which will be overridden.

- Is setState() method inherited from React.Component class?
- Yes.

- JavaScript has method overloading functionality, Yes or No and Why?
- No, It has only overriding functionality.
- In a language like java, for instance, the compiler will check the number and types of parameters passed to a function and match it with the function signature. In JavaScript however, type checking of parameters doesn't happen at compile time. In fact, the parameters won't be type checked even at run time unless they are actually used, and even then the type checking is extremely relaxed.



Contents

- ✓ OOPS Introduction
- ✓ Objects
- ✓ Classes
- ✓ Encapsulation
- ✓ Abstraction
- ✓ Inheritance
- ✓ Polymorphism
- ➢ **Summary**

Summary

1. Objects means simply real world entities such as mobile, laptop, bike, car, cat etc.,
2. Class is a blueprint for creating objects. We can create objects in different ways in JavaScript by using functions, object literals etc.,
3. **Encapsulation** is a method to hide the data in a single entity or unit along with a method to protect information from outside.
4. **Abstraction** is the method that "shows" only essential attributes and "hides" unnecessary information.
5. Inheritance is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods.
6. **Polymorphism** means having many forms.
7. JavaScript supports only overriding functionality because there is no type checking functionality.

Sharing is caring

SUBSCRIBE

LIKE

SHARE

COMMENT

Thank you for watching